SYSTEM AND METHOD FOR ACTIVELY MANAGING AN ENTERPRISE OF CONFIGURABLE COMPONENTS

Copyright Notice

Field of the Invention

The present invention relates in general to enterprise component management and, in particular, to a system and method for actively managing an enterprise of configurable components.

Background of the Invention

Enterprise computing environments include individual localized intranetworks interconnected via wide area internetworks or "Internets." The internetworks can include geographically distributed resources which, when taken as a whole, comprise a unified set of loosely-associated computers. The Internet is an example of a widely used public internetwork that can form part of an enterprise computing environment.

Most enterprise computing environments consist of heterogeneous computing systems, ranging from network appliances and personal computers to mainframes and specialized servers providing access to World Wide Web ("Web") content, Internet applications, and databases. To promote interoperability among these network components, configuration parameters must be carefully designated and tuned to maximize system performance and minimize areas of potential conflict and concern.

The ability to effectively manage inter-platform configurations is crucial to providing reliable and predictable performance. However, providing configuration management on an inter-platform basis is difficult. Different configurations of hardware, operating system, mid-tier and application software present a combinatoric maze of potential configuration parameter scenarios. Commonly-occurring combinations of inter-platform parameter configurations are often discovered by chance only after time-consuming trial and error.

As an example, a Unix operating system-based database server employing a shared memory model must be configured with a fixed number of buffers, shared memory pool space and processes to allow concurrent operation with multiple database instances. The operating system must also be configured to support an upper bound on shared memory size with a sufficient number of semaphores. The dependent relationships between the operating system configuration parameters and those mandated by the database shared memory model are critical to providing reliable concurrent database operation.

Many software vendors, such as Oracle Corporation, Redwood Shores, California, provide certified configurations that guarantee an acceptable level of performance and interoperability. Nevertheless, changes falling outside the bounds of certified configurations can propagate through an enterprise and cause a negative, and often unintentional, effect on dependent systems. Identifying dependent configuration parameter relationships is crucial to enabling impact analyses to be performed prior to effecting actual changes. Validating configuration parameters forms the core of impact analysis.

One problem in managing enterprise environments is the difficulty in checking dependent relationships *a priori.* The single system paradigm is widely accepted as a sufficient solution. For example, Windows-based environments employ a registry which records the installation of applications and components. The registry aids only individual system configuration management and does not allow the impact of a configuration parameter change be checked against the enterprise. Although dependent relationships may be implicit in the configuration

parameter values used, the dependent relationships themselves are not reflected on an inter-platform basis and cannot, therefore, be validated properly.

A further problem in managing enterprise environments is maintaining a consistent definition of "the truth," that is, the global master schemas for component, parameter, and node resource definitions. A relatd problem is enforcing the master configuration definitions against individual components operating within the enterprise.

In the prior art, four solutions attempt to provide inter-platform configuration management. These include the Infrastructure Management program, licensed by BMC, Houston, Texas; Tivoli Business System Manager, licensed by IBM, Armonk, New York; Platinum, licensed by Computer Associates, Islandia, New York; and AutoDBA, licensed by SenWare, Golden, Colorado. The Infrastructure Management program is composed of multiple modules which monitor and manage different parts of the infrastructure independent of each other. The Tivoli Business System Manager provides a single point of control for managing groups of applications across nodes and distributed systems. Platinum provides performance and event monitoring with user-defined job automation. AutoDBA utilizes neural networks for learned behavior. Although providing some form of inter-platform configuration management, the forgoing solutions fail to provide dependent relationship and context-based configuration management.

Therefore, there is a need for an approach to providing declarative enterprise-wide configuration validation and management. Preferably, such an approach would provide a configuration management operating from a common entry point on an internetwork and intranetwork basis.

There is a further need for an approach to providing a framework for enabling intra- and inter-platform software integration. Preferably, such an approach would identify and enforce configuration management policies through identified dependent relationships and parameter definitions.

Summary of the Invention

The present invention provides a system and method for dynamically managing configuration parameters applied in individual network components in an enterprise computing environment. A centralized management system includes a management configuration module that functions as the arbiter of all configuration parameter changes throughout a managed domain. A set of global parameter definitions and document type definitions are maintained in a database repository accessed by the management configuration module. Individual network components execute within the enterprise computing environment and are configured with configuration parameters as validated and managed by the management configuration module. Upon the installation or changing of a managed component, the configuration parameters are uploaded to the management configuration module as well-formed XML doclets. Each component is registered and each change is then validated, analyzed and managed. Validated changes are then propagated back to the individual network components as well-formed XML doclets and are chronicled in the database repository for future historical use and impact analysis.

One embodiment is a system and method for actively managing an enterprise of configurable components. A configuration for at least one individual component in a managed domain is stored. The configuration includes a set of configuration parameters and type definitions. A mapping of the configurations is defined between at least two of the individual components. Each mapping specifies a configuration parameter with a relationship dependent on at least one other such configuration parameter. Each such individual component is registered. The configuration parameters and type definitions are validated based on a master set of configuration parameters and type definitions defined for an enterprise. The mappings within the enterprise are enforced by dynamically probing the validated configurable parameters for each such individual component.

A further embodiment is a system and method for providing a framework for centrally managing configurations of distributed computing components. A

plurality of individual components are interfaced. Each component includes a client module applying document type definitions and storing configuration parameters. A database repository storing master document type definitions and global parameter definitions is maintained. The database repository is accessed and actively manages the individual components. The stored document type definitions for each individual component are retrieved via the corresponding client module. The configuration parameters are extracted from each retrieved document type definition with the master document type definitions. Each extracted configuration parameter is validated against the validated configuration parameters in the global parameter definitions.

Still other embodiments of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein is described embodiments of the invention by way of illustrating the best mode contemplated for carrying out the invention. As will be realized, the invention is capable of other and different embodiments and its several details are capable of modifications in various obvious respects, all without departing from the spirit and the scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive.

Brief Description of the Drawings

FIGURE 1 is a block diagram showing an enterprise computing environment, including a system for actively managing an enterprise of configurable components, in accordance with the present invention.

FIGURE 2 is a block diagram showing the system for actively managing an enterprise of configurable components of FIGURE 1.

FIGURE 3 is a data flow diagram of the core services provided by the management system of FIGURE 2.

FIGURE 4 is a block diagram showing the software modules of the management configuration module of FIGURE 2.

FIGURE 5 is a flow diagram showing a method for actively managing an enterprise of configurable components.

FIGURE 6 is a flow diagram showing the routine for initializing the management system for use in the method of FIGURE 5.

FIGURE 7 is a flow diagram showing the routine for processing a new configuration for use in the method of FIGURE 5.

FIGURE 8 is a flow diagram showing the routine for validating configuration parameters for use in the routine of FIGURE 7.

FIGURE 9 is a flow diagram showing the routine for processing unvalidated configuration parameters for use in the routine of FIGURE 7.

FIGURE 10 is a flow diagram showing the routine for processing a changed configuration for use in the method of FIGURE 5.

APPENDICES A-D include, by way of example, a sample document type definition (Appendix A), an XML doclet (Appendix B) and doclet fragments (Appendices C-D) formatted in accordance with the present invention.

Detailed Description

FIGURE 1 is a block diagram showing an enterprise computing environment 10, including a system for actively managing an enterprise of configurable components 11, in accordance with the present invention. The enterprise computing environment 10 comprises an intranetwork 12 interconnected to an internetwork 23, including the Internet, via a gateway 24 or similar bridging device. Other network topologies and configurations, and arrangements of clients, servers and components are feasible, as would be recognized by one skilled in the art.

The management system 11 configures, validates and manages the configuration parameters of a plurality of systems interconnected over the internetwork 12, including a Web cache 13, Internet application servers (iAS) 14-16, and database (DB) servers 17-18. The Web cache 13 provides dynamic caching of Web content for third party Web servers (not shown). The Internet application servers 14-16 flexibly execute Web site, portal and Internet applications to provide access to both traditional Web browsers and mobile devices. The database servers 17-18 are respectively coupled to relational databases 19-20 to provide management and access to structured data records. In

addition, the management system 11 can manage and configure remote systems, including a remote server 21 and remote client 22 interconnected via the internetwork 23.

The management system 11 automates and validates configuration changes to the runtime enterprise environment for the set of systems, including the Web cache 13, Internet application servers 14-16, and database servers 17-18 within a managed domain. The management system 11 provides a framework to enable the integration and management of system and user software from a common entry point, as further described below beginning with reference to FIGURE 2. The individual dependent relationships between the various enterprise components are managed by the management system 11 which utilizes a context-based impact analysis prior to validating proposed changes to the configuration parameters. Each managed component maintains a stored document type definition containing configuration parameters and values expressed as declarative definitions. In the described embodiment, the document type definitions are defined as doclets written in an Extensible Markup Language (XML) format.

The individual computer systems, including management system 11 and network managed components 13-18, 21, 22 are general purpose, programmed digital computing devices consisting of a central processing unit (CPU), random access memory (RAM), non-volatile secondary storage, such as a hard drive or CD ROM drive, network interfaces, and peripheral devices, including user interfacing means, such as a keyboard and display. Program code, including software programs and data, are loaded into the RAM for execution and processing by the CPU and results are generated for display, output, transmittal, or storage.

FIGURE 2 is a block diagram showing the system for actively managing an enterprise of configurable components of FIGURE 1. The management system defines a framework 30 that includes each of the individual network components 33, a management console 31 and a management server 32. Each individual network component 33 executes a client module (cbean) 37 on a Java virtual

machine (JVM) 36. The client modules 37 access stored configuration parameters 41. Each set of configuration parameters 41 stores the values of the tunable configuration parameters for operating system, mid-tier and application software executing on that particular network component 33.

In the described embodiment, the management console 31 executes as a Web browser-based application that presents an intuitive and user-friendly user interface. The core of the management system 30 is a management configuration (mconfig) module 35 that executes as a Java server application (servlet) on a Java virtual machine (JVM) 34. Alternatively, the management configuration module 35 could be a stand-alone application or implemented as a form of active server page, as would be recognized by one skilled in the art.

The management console 31 interfaces to a management server 32, coupled to a database repository 38. The database repository 38 includes a set of document type definitions (DTDs) 39 and a set of global parameters 42. Each document type definition 39 provides a master declarative definition of component parameters and dependent relationships for intra- and inter-platform software. Each DTD 39 identifies individual configuration parameters and defines a value type, value range and parameter reference relationships. In addition, the form of each dependent relationship is also defined and can include one-way, two-way, cyclic, one-to-many-to-one, and many-to-many relationships. Each global parameter defintion 42 provides a consistent master defintion of configuration parameter values to be used by each of the network components 33.

Operationally, the management configuration module 35 periodically probes the individual network components 33 for new or changed network components. The management configuration module 35 probes the individual network components 33 to discover new hardware resources, such as the number of processors, amount of physical random access memory, number of network interface cards, disk capacity, and so forth, to factor into the validation, impact analysis, and change control performed. In the described embodiment, the management configuration module 35 implements the public application programming interface (API) used by the Java Management Extension (JMX) to

invoke various services needed to collect, validate and analyze the individual configuration parameters 41.

Upon discovering a new or changed component or server, the management configuration module 35 requests and receives the runtime configuration parameters 41 from the corresponding client module 37. The configuration parameters 41 are received as well-formed XML doclets 40. Each XML doclet 40 conforms to the XML schema used by the management configuration module 35. The XML doclets 40 allow the management configuration module 35 to construct a component parameter relationship dependency tree for use in impact analysis. Only those modules where a change has been discovered will be validated in an established enterprise, to minimize network traffic.

The management configuration module 35 registers and validates each network component 33. The configuration parameters 41 are validated against the master document type definitions 39 maintained in the database repository 38. The master document type definitions 39 allow the supporting of certified configurations by enforcing the configuration parameters as a configuration policy. Upon the completion of validation, the management configuration module 35 generates and distributes a runtime XML doclet 40 that describes the resources used by each server in the enterprise computing environment 10 (shown in FIGURE 1).

FIGURE 3 is a process flow diagram 50 of the core services 51 provided by the management system 11 of FIGURE 1. The management system core services 51 include the following:

(1) Parameter validation check;

(2) Impact analysis prior to effecting a configuration parameter change;

(3) Consistent parameter change management based on impact analysis;

(4) Configuration version control;

(5) Restart and reboot from last valid configuration;

(6) Field configuration download service for new versions of document type definitions 39 (shown in FIGURE 2) and parameter definitions; and

(7) Field configuration upload to a centralized support site.

The core services 51 depend on four primary sources of input. First, pre-defined schemas for component and parameter definitions and node resource definitions 51 are retrieved by the management server 32 from the database repository 38 (shown in FIGURE 2). In addition, XML doclets containing site-specific fixed resources 53, site-specific components and parameters 54 and component registrations 55 are received from the individual network components 33 (shown in FIGURE 2).

The core services 51 obtain the site-specific resource and component configuration XML doclets 53-55 by invoking probing service layers (not shown). The core services 51 analyze, advise, and correct out-of-bound and invalid configuration parameters and identify cyclic relationships. A set of new XML doclets 58 are generated and fielded to the individual network components 33 for implementation. The core services 51 also exports management, advising and alert services 56. The core services 51 can be exported through a browser service 57, as implemented on the management console 31 (shown in FIGURE 2).

The attached Appendices include, by way of example, a sample document type definition (Appendix A), an XML doclet (Appendix B) and doclet fragments (Appendices C-D) formatted in accordance with the present invention for a hypothetical system having two processors, two gigabytes of random access memory, one network interface card, and 64 gigabytes of local disk space divided into four separate 17.2 gigabyte disk devices.

Appendix A shows a sample document type defintion for a managed node and resources in terms of specific hardware and software resources. The document type defintion is used to generate a well-formed XML doclet.

Appendix B shows the XML doclet generated from the document type definition of Appendix A. The doclet includes a DOCLET_VERSION and a DOCLET_DATE tag to allow multiple versions of the doclet to be maintained.

The XML tags and attributes allow load resources to be defined for an unlimited number of nodes within a managed domain. For efficiency, each doclet can be stored in a database repository in XML format.

When resources are configured on a given server node, the GLOBAL_NODE_DOCLET tag is referenced to determine if the resource exists and the available capacity, thereby preventing the overcommitment of resources. If the resource does not exist, the component configuration parameter is marked as invalid, thereby preventing the nonexistent resource from being configured.

Appendix C shows a sample doclet fragment that includes a declarative COMPUTED_VALUE tag. This type of doclet tag allows dependent parameter relationships to be dynamically managed by expressly defining dependent configuration parameter relationships as a computable value. The COMPUTED_VALUE tag includes an equation "1.1 * :R_001," which specifies the parameter definition for SESSIONS based on the number of PROCESSES.

Appendix D shows a sample doclet fragment illustrating intra-platform dependent relationships. By way of example, if the PROCESSES parameter is set to equal 300, the SESSIONS parameter is set to 330. As well, the REF_PARAMETER tag provides the component type, name and node reference for formulating the value of the SESSIONS tag.

Other forms of doclets, including various doclet implementations and other uses of scripting or non-scripting languages, and various tag definitions and arrangements, are feasible, as would be recognized by one skilled in the art.

FIGURE 4 is a block diagram showing the software modules 60 of the management configuration module 35 of FIGURE 2. The management configuration module 35 includes two main components: configuration, validation and management 60 and component-specific adapters 61. The configuration validation management 60 is composed of the following parts:

(1) Adapters 65: Used for reading and writing component-specific parameters received through the client modules 37 (shown in FIGURE 2) and via an Internet application server (iAS) system

management interface (SMI) classes to read and write specific parameter files.

(2) DOM XML parser 66: Reads and extracts XML doclet fragments from XML doclets 40.

(3) Configuration validater and analyzer 67: Ensures runtime configuration parameters are within operating values as specified by retrieved document type definitions 39 (shown in FIGURE 2).

(4) Service interfaces 68: Registers, validates and retrieves configuration parameters 41 from individual network components 33 (shown in FIGURE 2).

The configuration validation management component 60 can include multiple adapters 65 and service interfaces 68 to accommodate various forms of network components.

The component-specific adapters 61 read and write component-specific parameter file formats for heterogeneous network components 33.

The configuration validation management component 60 accesses stored global parameter definitions 62, node resources 63, and component-specific parameter values 64, all stored as well-formed XML doclets. The global parameter definitions 62 and node resources 63 are maintained in the database repository 38 accessible by the management server 32. Each proposed specific parameter value set 64 is retrieved from the individual network component 33 and stored in the database repository 38 for history and impact analysis.

In the described embodiment, the management configuration module 35 can be invoked either via interactive access or automated processing. Interactive access includes Web browser-based access and stand-alone programs that allow system administrators and other authorized users to view, change history, and, if authenticated, allow changes to the managed systems. When invoked via automation, the only service provided is component validation dependency checks to ensure that the pre-arranged configuration parameters are valid and that the dependent relationships hold true. The management configuration module 35 can

be invoked via automation whenever a new network component or server is installed or deployed, or when run on a periodic basis.

Each module is a computer program, procedure or module written as source code in a conventional programming language, such as the C++ programming language, and is presented for execution by the CPU as object or byte code, as is known in the art. The various implementations of the source code and object and byte codes can be held on a computer-readable storage medium or embodied on a transmission medium in a carrier wave. The accelerator system 61 operates in accordance with a sequence of process steps, as further described below with reference to FIGURE 5.

FIGURE 5 is a flow diagram showing a method for actively managing an enterprise of configurable components 80, in accordance with the present invention. The method first initializes the management system 30 (shown in FIGURE 2) (block 81), as further described below with reference to FIGURE 6. The management system 11 then enters an iterative and continuous processing loop (blocks 82-78), which ends when the management system 11 is terminated or shut down.

During each iteration (block 82), all network components and servers in the managed domain are probed (block 83) by requesting each client module 37 to identify, register and validate any new or changed components or servers. If a new component or server is discovered through probing (block 84), the new configuration is processed (block 85), as further described below with reference to FIGURE 7. Otherwise, if a changed configuration is discovered (block 86), the changed configuration is processed (block 87), as further described below with reference to FIGURE 10. Iterative processing proceeds (block 78) continuously until the system is terminated or shut down.

FIGURE 6 is a flow diagram showing the routine 90 for initializing the management system 11 for use in the method of FIGURE 5. The purpose of this routine is to download the necessary support files for use by the management system 11 prior to the initiation of the iterative processing loop.

Thus, the XML document type definitions 39 (shown in FIGURE 2) are downloaded from the database repository 38 (block 91). The XML formatted parameter definitions, stored as global parameter definitions 42 (shown in FIGURE 2) for all supported components, are also downloaded from the database repository 38 (block 92). The routine then returns.

FIGURE 7 is a flow diagram showing the routine 100 for processing a new configuration for use in the method of FIGURE 5. The purpose of this routine is to register and validate a new network component upon discovery through active probing.

Thus, the component configuration of each newly discovered component is registered on the management server 32 (block 101). Each individual component within the newly discovered configuration is iteratively processed (blocks 102-114) as follows. During each iteration (block 102), the runtime configuration parameters 41 (shown in FIGURE 2) for each component are recorded into a well-formed local XML doclet (block 103). The XML doclets are downloaded to the management server 32 (block 104) and are registered (block 105). The management configuration module 35 will wait (block 106) until registration is complete (block 105), after which the configuration parameters are validated (block 107), as further described below with reference to FIGURE 8.

If the configuration parameters 41 are not validated (block 108), the unvalidated configuration parameters are processed (block 109), as further described below with reference to FIGURE 9. Otherwise, if validated (block 108), the runtime version of the configuration parameters 41 are recorded into XML doclets 40 and assigned a version control number (block 110).

In the described embodiment, the DOCLET_VERSION and DOCLET_DATE tags are used as a version control number. Next, the XML doclet 40 is sent to the specific server for which the component configuration is defined (block 111). The specific server then ensures the configuration parameters 41 are applied by the client module 33. The runtime configuration parameters are stored into the database repository 38 by the management server 32 (block 112) and, as an optional step, uploaded to a designated support site

(block 113). Processing continues (block114) for each remaining component (block 102), after which the routine returns.

FIGURE 8 is a flow diagram showing the routine 120 for validating configuration parameters. The purpose of this routine is to validate, analyze and manage configuration parameters 41 (shown in FIGURE 2).

Thus, the configuration parameters 41 are validated against the XML document type definitions 39 (shown in FIGURE 2) retrieved from the database repository 38 (block 121). Validation includes checking the value type, value range and any parameter reference relationships. Next, an impact analysis is performed (block 122). The impact analysis includes dynamically traversing the dependent relationship tree to ensure that the configuration policy is enforced. If the configuration parameters reflect changed values (block 123), change management is performed (block 124). Change management includes managing, advising, and alerting a system administrator as to the effect of each change, as well as actually applying the changes. The routine then returns.

FIGURE 9 is a flow diagram showing the routine 130 for processing unvalidated configuration parameters for use in the routine of FIGURE 7. The purpose of this routine is gracefully process configuration parameters 41 (shown in FIGURE 2) that fail validation.

The unvalidated configuration parameters are processed by performing one or more selected operations (blocks 131-136), as follows. First, an unvalidated but changed configuration parameter can be changed back (132) to a previously-validated value or to a default value as indicated in the global parameter definitions 42 (shown in FIGURE 2). Second, the management system 11 can perform an audit (block 133) over all of the configuration parameters to determine the effect of the unvalidated configuration parameter. Third, an alert can be generated to the administrator (block 134). As well, the unvalidated configuration parameter change can be logged (block 135), thereby resulting in the acquiescence of the unvalidated configuration parameter change.

Upon completion of one or more of the foregoing operations, the routine returns.

FIGURE 10 is a flow diagram showing the routine 140 for processing a changed configuration for use in the method of FIGURE 5. The purpose of this routine is to register and validate a changed component upon discovery through probing.

Thus, the registration of the component configuration of the changed network component is obtained from the management server 32 (block 141). Each individual component within the newly discovered configuration is iteratively processed (blocks 142-152) as follows. During each iteration (block 142), the runtime configuration parameters 41 (shown in FIGURE 2) for each component are recorded into a well-formed local XML doclet (block 143). The XML doclets are downloaded to the management server 32 (block 144) and validated (block 145), as further described below with reference to FIGURE 8.

If the configuration parameters 41 are not validated (block 146), the unvalidated configuration parameters are processed (block 147), as further described below with reference to FIGURE 9. Otherwise, if validated (block 146), the runtime version of the configuration parameters 41 are recorded into XML doclets 40 and assigned a version control number (block 148).

In the described embodiment, the DOCLET_VERSION and DOCLET_DATE tags are used as a version control number. Next, the XML doclet 40 is sent to the specific server for which the component configuration is defined (block 149). The specific server then ensures the configuration parameters 41 are applied by the client module 33. The runtime configuration parameters are stored into the database repository 38 by the management server 32 (block 150) and, as an optional step, uploaded to a designated support site (block 151). Processing continues (block 152) for each remaining component (block 142), after which the routine returns.

While the invention has been particularly shown and described as referenced to the embodiments thereof, those skilled in the art will understand that the foregoing and other changes in form and detail may be made therein without departing from the spirit and scope of the invention.